LIST REKURSIF

Defri Kurniawan
defri.kurniawan@dsn.dinus.ac.id

Struktur Data 2016

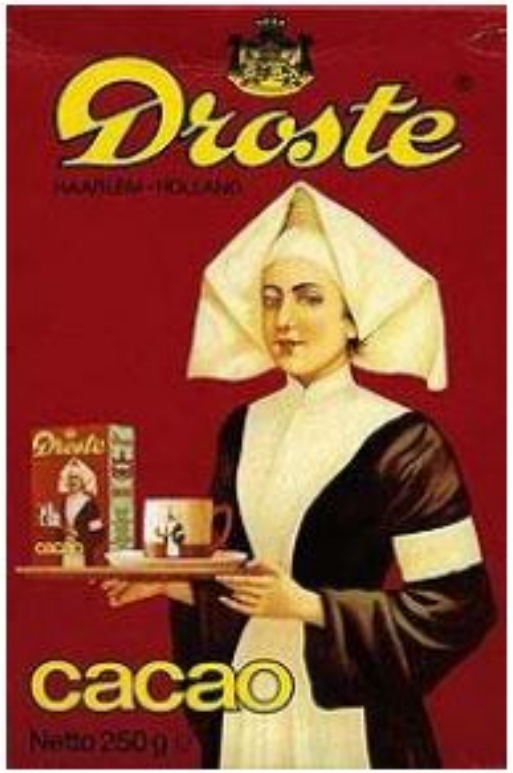
RENCANA KEGIATAN PERKULIAHAN SEMESTER

W	Pokok Bahasan
1	ADT Stack
2	ADT Queue
3	List Linear
4	List Linear
5	List Linear
6	Representasi Fisik List Linear
7	Variasi List Linear
8	**Ujian Tengah Semester**

W	Pokok Bahasan
9	Variasi List Linear
10	Double Linked List
11	Stack dengan Representasi List
12	Queue dengan Representasi List
13	List Rekursif
14	Pohon dan Pohon Biner
15	Multi List
16	**Ujian Akhir Semester**

Rekursifitas

- Suatu entitas disebut **rekursif** jika pada definisinya terkandung dirinya sendiri
- Program prosedural juga dapat bersifat rekursif
- Aspek-aspek pemrograman prosedural yang dapat bersifat rekursif: *prosedur, fungsi, struktur data*
  - Program utama tidak dapat bersifat rekursif karena tidak berparameter



Mengingat kembali:
Analisis Rekurens

- Analisis rekurens: penalaran berdasarkan definisi rekursif
- Teks program rekursif terdiri dari dua bagian:
  - **Basis** (Basis-0 atau Basis-1): menyebabkan prosedur/fungsi berhenti
  - **Rekurens** : mengandung call terhadap prosedur/fungsi tersebut, dengan parameter bernilai mengecil (menuju basis)

Studi Kasus: Faktorial

- Definisi – 1:

  0! = 1 { basis }

  N! = N * (N-1)! { rekurens }

- Definisi – 2:

  1! = 1 { basis }

  N! = N * (N-1)! { rekurens }

- Implementasi kasus faktorial ke dalam fungsi/prosedur rekursif dari sudut pandang:
  1. Terjemahan dari program fungsional rekursif
  2. Terjemahan sebuah loop (iteratif) menjadi rekursif (definisi faktorial ditransformasi menjadi perhitungan deret kali)

1. Terjemahan dari Program Fungsional Rekursif

- Mengingat kembali: Fungsi Factorial (Fungsional)

```
DEFINISI DAN SPESIFIKASI
factorial : integer ≥ 0 → integer > 0
{ factorial(N) = N! sesuai dengan definisi rekursif factorial }
```

```
REALISASI (VERSI-1)
{ Realisasi dengan definisi factorial sebagai berikut jika
factorial(N) adalah N!:
                N = 0 : N! = 1
                N ≥ 1 : N! = N * (N-1)! }
factorial(N) : if N = 0 then { Basis 0 }
                      1
               else { Rekurens : definisi faktorial }
                      N * factorial(N-1)
```

2. Terjemahan dari loop (iteratif) menjadi program rekursif

- Dalam konteks prosedural kita memiliki “loop” sebagai mekanisme untuk mengulang
- Kita dapat mensimulasi mekanisme pengulangan secara rekursif:
  - Parameter hasil dan variabel temporary pada mekanisme pengulangan dipindahkan menjadi parameter prosedur
  - Hati-hati dalam meletakkan nilai inisialisasi untuk parameter input dan input/output

Fungsi Factorial (Iteratif) - 1

```
function factorial1 (N : integer) → integer
{ Mengirim N! sesuai dengan definisi faktorial:
1*1*2*3*4*...(N-1)*N }
{ Pada persoalan ini, definisi rekurens faktorial
ditransformasi menjadi perhitungan deret kali }
```

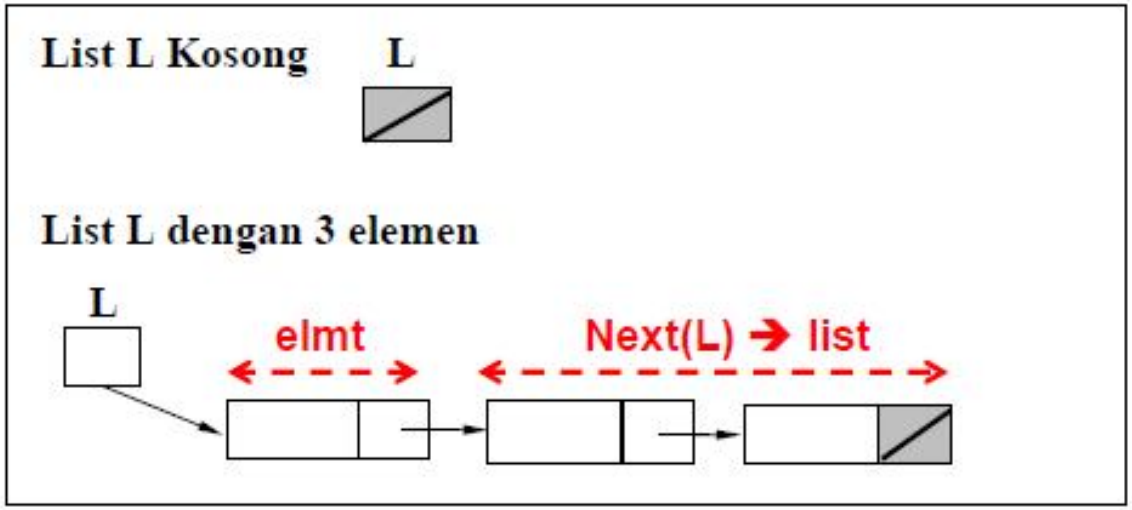
```
KAMUS LOKAL
        Count : integer
        F : integer
```

```
ALGORITMA
   F ← 1          { inisialisasi }
   Count ← 1      { first element }
   while (Count <= N) do
      F ← F * Count          { Proses }
      Count ← Count + 1      { Next element }
   { Count > N, semua sudah dihitung }
   → F { Terminasi }
```

Pemrosesan List Linier secara Prosedural - Rekursif

List sebagai Struktur Data Rekursif

- Definisi rekursif list linier:
  - **Basis**: list kosong adalah list
  - **Rekurens**: list tidak kosong terdiri atas sebuah elemen dan sisanya adalah list



Struktur Data List untuk Pemrosesan secara Rekursif (Notasi Algoritmik)

```
KAMUS
{ List direpresentasi dg pointer }
      type infotype : ... { terdefinisi }
      type address  : ... { terdefinisi }
      type ElmtList : < info : InfoType,
                        next : address >

      type List : address

{ Deklarasi  nama untuk variabel  kerja }
      L : List
      P : address   { address untuk traversal }
{ Maka penulisan First(L) menjadi L
  Next(P), Info(P) tergantung representasi fisik yang
  digunakan }
```

Struktur Data List untuk Pemrosesan secara Rekursif (Bahasa C, pointer)

```
#define Nil NULL

/* Selektor */
#define Info(P) (P)->info
#define Next(P) (P)->next

typedef int infotype;
typedef struct tElmtlist *address;
typedef struct tElmtlist {
        infotype info;
        address next;
} ElmtList;
/* Definisi list : */
/* List kosong : First(L) = Nil */

typedef address List;
```

Primitif Dasar: Pemeriksaan List Kosong

- Notasi Algoritmik (rep. berkait)

```
function  IsEmpty (L : List) → boolean
{ Tes apakah sebuah list L kosong.
  Mengirimkan true jika list kosong, false jika tidak kosong }
KAMUS LOKAL
ALGORITMA
    → (L = Nil)
```

- Bahasa C (rep. berkait dgn. pointer)

```
boolean IsEmpty (List L)
/* Tes apakah sebuah list L kosong.
Mengirimkan true jika list kosong, false jika tidak kosong */
{       /* Kamus Lokal */

        /* Algoritma */
        return (L == Nil);
}
```

- Selain pengecekan list kosong, primitif lain yaitu
  - PrintList() = untuk mencetak list
  - NbElmtList() = menghitung banyaknya list

Pemrosesan List Linier secara Rekursif

(Mengacu pada Diktat
"Pemrograman Fungsional")

Struktur Data List

- type List: [ ] atau [e o List]

- Primitif dasar:
  - Selektor: FirstElmt, Tail
  - Konstruktor: Konso, Kons•

Selektor

```
function FirstElmt (L : List) → infotype
{ Mengirimkan elemen pertama sebuah list L yang tidak kosong }
KAMUS LOKAL
ALGORITMA
        → Info(L)
```

```
function Tail (L : List) → List
{ Mengirimkan list L tanpa elemen pertamanya, mungkin yang
dikirimkan adalah sebuah list kosong }
KAMUS LOKAL
ALGORITMA
        → Next(L)
```

Konstruktor - Konso

```
function Konso (L : List, e : infotype) → List
{ Mengirimkan list L dengan tambahan e sebagai elemen pertamanya }
{ Jika alokasi gagal, mengirimkan L }
```

```
KAMUS LOKAL
    P : address
```

```
ALGORITMA
    P ← Alokasi(e)
    if (P = Nil) then
        → L
    else
        { Insert First }
        Next(P) ← L
        → P
```

Konstruktor - Kons•

```
function Kons• (L : List, e : infotype) → List
{ Mengirimkan list L dengan tambahan e sebagai elemen terakhir }
{ Jika alokasi gagal, mengirimkan L }

KAMUS LOKAL
    P : address
    Last : address

ALGORITMA
    P ← Alokasi(e)
    if (P = Nil) then
        → L
    else
       { Insert Last }
       if IsEmpty(L) then { insert ke list kosong }
           → L
       else
           Last ← L
           while (Next(Last) ≠ Nil) do
                Last ← Next(Last)
           { Next(Last)=Nil; Last adl. alamat elemen terakhir }
           Next(Last) ← P
           → L
```

TERIMA KASIH